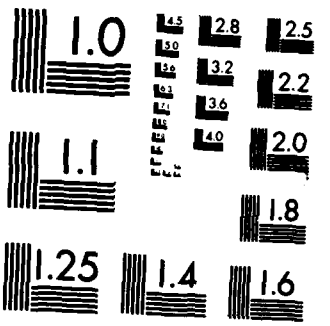
AD-A185 612 MICROWAVE LANDING SYSTEM (MLS) MATHEMATICAL MODEL INSTALLATION MANUAL(U) MSI SERVICES INC WASHINGTON DC J D JONES MAY 87 DOT/FAA/CT-TN87/17 DTFA03-86-C-00055 1/1

UNCLASSIFIED F G 12/5 NL

END
DATE FILMED
1-88



(2)

Microwave Landing System (MLS) Mathematical Model Installation Manual

Jesse D. Jones

May 1987

DOT/FAA/CT-TN87/17

U.S. Department of Transportation
Federal Aviation Administration

Technical Center
Atlantic City Airport, N.J. 08405

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/CT-TN87/17	AD-A185612	

4. Title and Subtitle	5. Report Date
MICROWAVE LANDING SYSTEM (MLS) MATHEMATICAL MODEL INSTALLATION MANUAL	May 1987
	6. Performing Organization Code: ACT-140
7. Author(s): Jesse D. Jones	8. Performing Organization Report No.: DOT/FAA/CT-TN/87/17
9. Performing Organization Name and Address: Department of Transportation, Federal Aviation Administration, Technical Center, Atlantic City International Airport, N.J. 08405	10. Work Unit No. (TRAIS)
	11. Contract or Grant No.: T0603N
12. Sponsoring Agency Name and Address: Department of Transportation, Federal Aviation Administration, Program Engineering and Maintenance Service, Washington, D.C. 20590	13. Type of Report and Period Covered: Technical Note, June 1986 - February 1987
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This Technical Note provides requirements and guidance for the installation of the baselined microwave landing system (MLS) mathematical model. This model originated at the Massachusetts Institute of Technology, Lincoln Laboratory, during the late 1970's, and has been revised and baselined to facilitate future software support and improvements. A majority of the work in developing this Technical Note was performed under contract DTFA03-86-C-00055, Task B-2 by:

MSI Services Incorporated
600 Maryland Avenue, S.W.
Suite 695
Washington, D.C. 20024

Changes included:

17. Key Words	18. Distribution Statement
Microwave Landing System / MLS / Mathematical Model / Installation Manual/MLS Mathematical Model	This document is available to the U.S. public through the National Technical Information Service, Springfield, Va. 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	25	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This Technical Note provides an estimate of system requirements and installation guidance for the baselined microwave landing system (MLS) mathematical model. This model was originally developed by the Massachusetts Institute of Technology, Lincoln Laboratory, during the late 1970's. Different users of the model subsequently obtained copies of the model with various levels of improvements. Therefore, to facilitate future software support and improvements, the Federal Aviation Administration (FAA) contracted to have the model baselined with all known improvements and to make the data input user friendly.

INTRODUCTION

PURPOSE.

The purpose of this manual is to provide the user with instructions that will permit the installation of the microwave landing system (MLS) Lincoln Laboratory simulation software at the user's computer facility. It is intended that the procedure for making revisions to the model will be accomplished by a reinstallation and, therefore, "installation" and "system update" will use the same procedure as described in this manual.

A general procedures section is included to give the user or the person installing the simulation a basic idea of the steps that are required in the installation process. This section will NOT provide system specific procedures because these will be different for each installation, i.e., the installation of the software on an IBM 3033 mainframe will require different commands and utilities than installation on a DEC VAX 11/780.

Appendixes have been furnished to provide some system specific examples of computer utilities and setups that could be employed to actually accomplish the installation at a given computer facility.

A specification for user supplied graphics software is included in appendix A.

INTENDED AUDIENCE.

The manual is written for someone with computer background. In order to implement the procedures described in this manual, the user will have to have access to the target computer system and be familiar with some of the system utilities (e.g., editors and compilers) that reside in the system. Some of the necessary information can probably be obtained from the system administrator, or technical staff if the facility happens to be a large one.

REQUIREMENTS

HARDWARE REQUIREMENTS.

The computer that is to serve as the target system must have the following hardware elements:

1. 1/2 inch 9-track 1600 bits per inch (bpi) tape drive
2. 1 megabyte applications program memory
3. Floating point processor (not required but recommended)
4. Graphical output device.

The model has also been compiled (using a FORTRAN-77 compiler) and run on an IBM personal computer (PC)/AT equivalent with the following hardware elements:

1. 1.2 megabyte floppy disk drive

2. 640k bytes of memory

3. Floating point processor

4. Plotter (graphical output device).

SOFTWARE REQUIREMENTS.

The model is distributed in source code form, therefore, the following utility programs must be available on the target system:

1. FORTRAN 77 (ANSI X3.9-1978) compatible compiler

2. Graphics library (Tektronix, CALCOMP, etc.)

3. User written graphics software to interface with the graphics library.

STORAGE REQUIREMENTS.

In order to install the model on your computer there must be sufficient direct access storage space available to hold the source code, object and load modules, and the sample input and output data sets.

The following is a breakdown of approximate storage requirements:

		Executable	
Program	Code	VAX 11/750	PC AT
BMLST	1200k bytes	83k bytes	250k bytes
BMLSR	280k bytes	47k bytes	125k bytes
BPLOTT	580k bytes	76k bytes	190k bytes
BPLOTR	365k bytes	96k bytes	160k bytes
Plotting files	400-1000k bytes		
Intermediate file (data from BMLST to BMLSR)	1-4M bytes		

Estimates for load modules will fluctuate on different systems and for different compilers. The byte count for source code statements is based on an 80-byte record. Some systems may be able to store these data in less space, but the transmittal tapes will always contain fixed 80-byte records. Compression may be possible if the compiler can accept statements that are less than 80 characters. Of course, the source can be removed after the compilation and link steps are completed.

DISTRIBUTION MEDIA AND LAYOUT

The simulation software and supporting data sets are contained on 1/2 inch 9-track 1600 bpi unlabeled magnetic tape.

The order of the data is as follows:

FILE 1	BMLST SOURCE (Propagation Model)
FILE 2	BMLSR SOURCE (System Model)
FILE 3	BPLOTT SOURCE (Propagation Model Plotting)
FILE 4	BPLOTR SOURCE (System Model Plotting)
FILE 5-N	SAMPLE PROGRAM INPUTS
FILE N+1...	OTHER FILES AS REQUIRED

All files are ASCII coded 80-character text records. The data will be blocked at 40 records to a physical block or 3200 bytes.

GENERAL PROCEDURES

Installation and checkout of the simulation software consists of the following steps:

1. Copy source code from distribution media to target machine disk using separate files for BMLST, BMLSR, BPLOTT, and BPLOTR.

2. Copy sample input data sets likewise.

3. Create user supplied graphics interface software, compile code, and create object file for subsequent linking. See "Graphics Interface" section.

4. Compile BMLST and link with SINIT and SOPEN creating a BMLST executable image. (Note: SINIT and SOPEN are provided with original installation tape for user adaptation of initialization parameters).

5. Repeat Step 4 for BMLSR.

6. Compile and link BPLOTT with SINIT, SOPEN, and USERPLOT. (Note: USERPLOT refers to the user written graphics software. The user can, of course, substitute any name here).

7. Repeat step 6 for BPLOTR.

8. Execute BMLST with the sample input files. This may require building installation dependent job control language (JCL) to execute and relate physical files with logical unit numbers.

9. Execute BPLOTT to obtain plots and compare these with sample plots received with original installation.

10. Repeat steps 8 and 9 with BMLSR and BPLOTR.

11. If the model fails to complete execution or produces strange results not matching the sample outputs, save the output and consult with your installation technical staff for probable cause of the error. Report the failure and any known causes to the Guidance and Airborne Systems Branch, ACT-140, FAA Technical Center, Atlantic City International Airport, N.J. 08405.

12. Create a "secure" backup copy of all installed software.

ABOUT THE APPENDIXES

The following appendixes describe in some detail the required graphics interface software and the necessary computer setups for installing the simulation software and sample input/output data sets.

The sample installation procedures have been written in as "generic" a form as is possible. All computer installations tend to tailor their job control language for their installation. This means that the setups that are shown here as samples will require slight modification at any given site.

APPENDIX A

GRAPHICS INTERFACE SOFTWARE SPECIFICATIONS

GRAPHICS INTERFACE SOFTWARE SPECIFICATION

The microwave landing system (MLS) math model has been baselined to assure the conformity of code for all users. Every effort has been made to standardize the code to minimize adaptability problems to different computers. Therefore, in order to make the MLS math model more transportable between computers, calls to graphical subroutines have been minimized. The subroutines which will be needed are defined functionally below. The appropriate software to implement these functions should be developed by each user on-site to assure the desired graphical actions for your graphical display device or plotter when executing programs BPLOTT and BPLOTR. The subroutines SINIT and SOPEN are required by all four programs of the MLS math model. Most of the remaining graphical subroutines called will be similar to standard CALCOMP plotter subroutines in the actions performed.

CALL SINIT - Will provide site specific data, constants, and time/date. This subroutine will be called by all four model programs and defines the computational environment.

CALL SOPEN - Will open file for input/output as appropriate.

CALL SIPLOT - Will initialize and select output device. This subroutine is used by BPLOTT and BPLOTR to define and specify the plotting or graphical environment.

CALL SNPLOT - Will reassign a new origin for the axes.

CALL SAXIS - Will plot axis and labels.

CALL SNUMBR - Will draw a number for plot annotation, etc.

CALL STITLE - Will draw title, label, etc.

CALL SLINE - Will produce a plot of data X,Y pairs.

CALL SSYMBL - Will draw a single symbol.

CALL EXTPLT - Exit plotting system.

CALL SCALIU - Convert inches to "user units."

CALL SCALUI - Convert "user units" to inches.

CALL SSCALE - Will examine data arrays and determine scaling values (not currently used).

The specific details of how these will be implemented are listed on the following pages. In addition, the user may need to write other additional subprograms to implement the above functions. It is also important to note that, in some cases, it may not be possible to implement an option, such as rotated labels, due to limitations of your equipment. In those situations, a suitable alternative should be implemented.

SINIT SUBROUTINE.

The SINIT subroutine is used to initialize common areas with site specific data such as installation, location, and constants (including date and time) constrained by machine capabilities. Therefore, only one call will be made to this subroutine in each program of the model.

There will be no arguments for the calling sequence:

CALL SINIT

All data will be passed from this subroutine in COMMON blocks. These COMMONS are:

1. /TITLE/ consisting of variables:

   IRID, a 4-character run identification (initialized by input file).

   RDATE, a 9-character date (initialized by this subroutine). RDATE is a 9-character date of the form DD-MMM-YY derived from the system date function. Some coding may be necessary to put the date in this format.

   RTIME, an 8-character time (initialized by this subroutine). RTIME is an 8-character representation of the time in the form HH:MM:SS. Seconds are unnecessary if not readily available from a system call.

   TITLE, a 60-character title field (initialized by input file).

2. /SITE/ consisting of variables:

   NAME, a 40-character field (initialized by this subroutine) to identify the installation or user performing the modeling.

   ADDR, a 40-character field (initialized by this subroutine) identifying the location or address of the above installation.

3. /IODEV/ consisting of variables (initialized in this subroutine):

   ISIU, an integer variable identifying the unit number for manual program input (terminal, tty, etc.)

   ISOU, an integer variable identifying the unit number for output associated with the above input terminals.

4. /MLSTIO/ consisting of integer variable (initialized in this subroutine):

   PID, the unit number from which the MLST input data (formatted input file) is obtained.

5. /MLSRIO/ consisting of integer variables (initialized in this subroutine):

   RID, the unit number from which the MLSR input data is obtained (BMLST output, original LL file 8).

   RDO, the BMLSR confidence counter and flag output data file.

6. /PMLSTD/ consisting of integer variables (initialized in this subroutine):

   PPID, the unit number for the input data to be plotted as MLST diagnostic plots (BMLST graphical output data).

   PPOD, the unit number used for the plotter or other display device.

7. /PMLSRD/ consisting of integer variables (initialized in this subroutine):

   PRID, the unit number for the input data (original LL file 12) to be plotted as error data.

   PROD, the unit number used for the plotter or other display device.

8. /MFPIO/ consisting of integer variables (initialized in this subroutine):

   MFID, the unit number for measured flight path input data.

SUBROUTINE SOPEN.

The SOPEN subroutine is used to open files for input/output. This subroutine is called once for each file to be opened.

The calling sequence

CALL SOPEN (LDEV, NAME, STAT, RLEN)

has four arguments as follows:

LDEV: an integer variable, is the logical device number to be associated with the input/output file. This number will be one of the variables assigned a value by SINIT and will be supplied to SOPEN by the subroutine call.

NAME: a character variable of length 40 bytes specifying the file name to be associated with LDEV. This file name is specified by the user during the question and answer sequence when the program is run.

STAT: a character variable of length 3 bytes specifying the status of the file being opened. STAT will be given a value of either 'NEW' or 'OLD' by the call to SOPEN. The user can change the value of STAT within SOPEN, if necessary.

RLEN: an integer variable indicating the record length of the file being opened. RLEN will be given a value in the call to SOPEN. The user can change the value of RLEN in SOPEN, if necessary.

This subroutine should include the labeled common blocks MLSTIO, MLSRIO, PMLST, PMLSRD, and MFPIO which contain the variables initialized by SINIT to the appropriate logical device numbers. These will be passed to SOPEN along with an appropriate status and record length and the file name supplied by the user interactively. It is suggested that SOPEN contain error traps to allow the user to change the file name and/or the status without terminating the program.

If the operating system requires that file names and logical device numbers be assigned externally to the program (i.e., JCL, etc.), this subroutine may exist without internal coding so that the driver programs will execute properly.

This subroutine will be a companion to subroutine SINIT and must be included wherever SINIT is used.

Record lengths used by the various programs are expected to be as follows:

BMLST

PID: formatted input file = 80 bytes
RID: multipath output data = 132 bytes
PPID: plot output data = 97 bytes
MFID: measured flightpath input data = 55 bytes

BMLSR:

RID: multipath output data = 132 bytes
RDO: flag output data = 44 bytes
PRID: plot output data = 131 bytes

SIPLOT SUBROUTINE.

The SIPLOT subroutine is used to initialize the plotting system. It will be called only once. This subroutine sets up any necessary plot system constants and opens the plot output device by performing standard file-opening procedures as necessary through the computer's operating system.

Additional data to be initialized by this subroutine are variables XINCH and YINCH which make up common /PLTSIZ/. XINCH is the x-distance (in inches) between the origin and the right edge of the plotting surface. YINCH is the y-distance (in inches) between the origin and the top edge of the plotting surface. These values are used for clipping of the output.

The calling sequence

CALL SIPLOT(LDEV)

has one argument:

LDEV: is the logical output device number, i.e., the device number of the plotter if it is on-line or the logical number of an output device for off-line plotting (PPOD or PROD initialized in subroutine SINIT).

Tables generated by BPLOTT will be written to ISOU. Therefore, tables may be redirected to a printer or other device by redirecting ISOU in this subroutine. This subroutine is called after all program dialog with tne user. Therefore, the redirection by this subroutine will only have an impact on the tables or any error messages generated by the system.

SNPLOT SUBROUTINE.

The SNPLOT subroutine is used to reassign the origin for the X and Y axes and/or to perform a page advance (clear screen) or whatever command is needed to provide a clean surface for the preparation of a new plot. The calling sequence

CALL SNPLOT (IDEV)

has one argument:

IDEV: is the logical output device number to which the current output is being directed, i.e., tables will be directed to ISOU, whereas plots will be directed to PPOD or PROD.

This subroutine is called with IDEV=ISOU for outputting of tables, and IDEV=PPOD or PROD when displaying plots. Thus, the user may develop the software for this subroutine to advance the printer or plotter as required.

SAXIS SUBROUTINE.

Most graphs require axis lines and scales to indicate the orientation and values of the plotted data points. The type of scaled axis produced by the SAXIS subroutine is a line at 0^{o} or 90^{o} angle, (X or Y) divided into one-inch segments. This call also annotates the divisions with appropriate scale values and labels the axis with a centered title. SAXIS is called separately for each X and Y axis.

The five arguments in the calling sequence are as follows:

CALL SAXIS (AXIS, LABL, NCHAR, FIRSTV, DELTAV)

AXIS: is a character variable (X or Y) specifying which axis is to be drawn. The axis length will be 9 inches for X and 6 inches for Y assuming an 8 1/2 x 11 inch plotting surface. Otherwise, the X-axis should be centered and occupy 9/11ths of the plot X-dimension. The Y-axis would then span a ratio 6/8.5 of the Y-dimension beginning 1/8.5 from the bottom of the plot.

LABL: is the axis label, which is centered and placed parallel to the axis line. This parameter is a character variable of not more than 40 characters. All annotation appears on the negative (clockwise) side of the X-axis. All annotation appears on the positive (counterclockwise) side of the Y-axis.

NCHAR: is the number of characters in the label.

FIRSTV: is the starting value which will appear at the first tick mark (origin) on the axis. This value will be specified by the model input data file in order to standardize plotting scales.

DELTAV: represents the number of data units per inch of axis. This value (increment or decrement), which is added to FIRSTV for each succeeding 1-inch division along the axis, will also be specified in the model input data file.

SNUMBR SUBROUTINE.

SNUMBR converts a floating-point number to the appropriate decimal equivalent so that the number may be plotted in the FORTRAN F-type format.

The SNUMBR calling sequence has six arguments:

CALL SNUMBR (XPAGE, YPAGE, HEIGHT, FPN, ANGLE, ±NDEC)

XPAGE, YPAGE: are the coordinates of the number in user data units.

HEIGHT: is the height (and width), in inches, of the symbol to be drawn.

FPN: is the floating-point number that is to be converted and plotted.

ANGLE: is the angle in degrees from the X-axis, at which the annotation is to be plotted. If ANGLE=0.0, the number will be plotted right side up and parallel to the X-axis. Note: Implementation of rotated numbers at any angle may not be feasible with all graphics packages.

±NDEC: controls the precision of the conversion of the number FPN. If the value of NDEC greater than 0, it specifies the number of digits to the right of the decimal point that are to be converted and plotted, after proper rounding. For example, assume an internal value (perhaps in binary form) of -0.12345678×10^3. If NDEC were 2, the plotted number would be -123.46.

STITLE SUBROUTINE.

The STITLE subroutine produces plot annotation at any angle and in practically any size. This call can be used to draw text such titles, captions, and legends.

The STITLE calling sequence has six arguments:

CALL STITLE (XPAGE, YPAGE, HEIGHT, CHAR, L, ANGLE)

XPAGE, YPAGE: are the coordinates, in inches, of the lower left-hand corner (before character rotation) of the first character to be produced. The pen is up while moving to this point.

HEIGHT: is the height, in inches, of the character to be plotted. The plotting routines assume the width of a character, including spacing, is normally the same as the height (e.g., a string of 10 characters, 0.14 inch high is 1.4 inches wide).

CHAR: is the text, in character representation to be used for annotation. The character(s) must be contiguous in a single variable. This variable is to be dimensioned for 60 characters.

L: the number of characters in the title.

ANGLE: is the angle, in degrees from the X-axis, at which the annotation is to be plotted. If ANGLE = 0.0, the character(s) will be plotted right side up and parallel to the X-axis. Note: implementation of rotated titles at any angle may not be feasible with all graphics packages.

SLINE SUBROUTINE.

The SLINE subroutine produces a line plot of the pairs of data values in two arrays (X and Y). The line produced may be either continuous or dashed. SLINE computes the page coordinates of each plotted point according to the data values in each array and the respective scaling parameters. The data points may be represented by centered symbols and/or connecting lines between points.

The calling sequence has five arguments:

CALL SLINE (XARRAY, YARRAY, ±NPTS, ±LINTYP, CHAR)

XARRAY: is the name of the array containing the abscissa (X) values in user data units.

YARRAY: is the name of the array containing the coordinate (Y) values in user data units.

±NPTS: is the number of data points in each of the two arrays just mentioned. If NPTS is negative, a dashed line is produced. The number of points in each array must be the same. The MLS model plotting programs will limit the number of points to 3000. However, the associated arrays will be dimensioned to 3002 to permit scaling factors needed by CALCOMP plotters to be added to the arrays.

±LINTYP: is the control parameter which describes the type of line to be drawn through the data points. The magnitude of LINTYP determines the frequency of plotted symbols, e.g., if LINTYP = 4, a special symbol (denoted by CHAR) is plotted at every fourth data point. If LINTYP is zero, the points are connected by straight lines but no symbols are plotted. If LINTYP is positive, a straight line connects every data point defined in the array. (The pen is up when moving from its current position to the first point.) If LINTYP is negative, no connecting lines are drawn; only the symbols are plotted.

CHAR: is the character equivalent of the special plotting symbol desired at each data point. If possible, this character should be centered.

SSYMBL SUBROUTINE.

The SSYMBL subroutine produces a single symbol based on the index value of INTEQ. The SSYMBL calling sequence

CALL SSYMBL (XPAGE, YPAGE, HEIGHT, CHAR, CDUF)

consists of five arguments:

XPAGE, YPAGE: are the coordinates of the symbol in user data units. It is desirable that XPAGE, YPAGE represent the geometric center of the character produced.

HEIGHT: is the height (and width), in inches, of the symbol to be drawn.
CHAR: is the character equivalent of the desired symbol.

CDUF: is a character argument which determines whether the pen is up or down during the move to XPAGE, YPAGE.

Where:

U = the pen is up during the move, after which a single symbol is produced.

D = the pen is down during the move, after which a single symbol is produced.

EXTPLT SUBROUTINE.

The EXTPLT subroutine is executed to exit the plotting programs. The calling sequence:

CALL EXTPLT

has no arguments, and will be the last call in the plotting driver program. This subroutine will be used to close files and plotting devices. Any device numbers necessary to close the users plotting system should be passed into this subroutine by the use of labeled common areas initialized and/or defined by SIPLOT.

SCALIU SUBROUTINE.

The SCALIU subroutine inputs an X, Y coordinate pair in inches and provides an equivalent X, Y coordinate pair in "user data units." The calling sequence:

CALL SCALIU (XI, YI, X, Y)

consists of the arguments:

XI, YI: X and Y coordinate pair in inches referenced to the plot origin.

X, Y: equivalent X-Y location in "user data units."

SCALUI SUBROUTINE.

The SCALUI subroutine takes an X, Y coordinate pair in "user data units" and provides the equivalent coordinates in inches. The calling sequence:

CALL SCALUI (X, Y, XI, YI)

consists of the arguments:

X, Y: X and Y coordinate pair in "user data units"

XI, YI: equivalent location in inches referenced to plot origin.

SSCALE SUBROUTINE. (Not currently used)

Typically, the user's program will accumulate plotting data in two arrays:

An array of independent variables, X_i

An array of dependent variables, $X_i = f(X_i)$

It would be unusual if the range of values in each array corresponded exactly with the number of inches available in the actual plotting area. For some problems the range of data is predictable. The MLS math model has predetermined standard scale factors for use in drawing the axes and plotting the data points on the graph. For special applications, however, these factors may not be known in advance.

Therefore, the SSCALE subroutine is used to examine the data values in any array and to determine a starting value (FIRSTV, minimum or maximum) and a scaling factor (DELTAV, positive or negative) such that:

1. The scale annotation drawn by the SAXIS subroutine at each division will properly represent the range of real data values in the array.

2. The data points, when plotted by the SLINE subroutine, will fit in a given plotting area. These two values are computed and returned by the call to SSCALE.

The scaling factor (DELTAV) that is computed represents the number of data units per inch of axis, but is adjusted so that there is always an interval of 1, 2, 4, 5, or 8×10^n (where n is an exponent consistent with the original unadjusted scaling factor). Thus, an array may have a range of values from 301 to 912, to be plotted over an axis of 10 inches. The unadjusted scaling factor is (912-301)/10 = 61.1 units/inch. The adjusted DELTAV would be $8 \times 10^1 = 80$.

The starting value (FIRSTV), which will appear as the first annotation on the axis, is computed as some multiple of DELTAV that is equal to or outside the limits of the data in the array. For the example given above, if a minimum is wanted for FIRSTV, 240 would be chosen as the best value. If a maximum is desired instead, 960 would be selected. In some instances, FIRSTV is selected as a downward-rounded value of the lowest actual data, and in other instances, the DELTAV ia adjusted upwards. An attempt is then made to center the data.

There are six arguments in the calling sequence:

CALL SSCALE (ARRAY, AXLEN, NPTS, ±INC, FIRSTV, DELTAV)

ARRAY: is the first element of the array of data points to be examined.

AXLEN: is the length of the axis (in inches) to which the data is to be scaled. Its value must be greater than 1.0 inch.

NPTS: is the number of data values to be scanned in the array.

±INC: is an integer whose magnitude is used by SSCALE as the increment with which to select the data values to be scanned in the array.

Normally INC = 1; if it is 2, every other value is examined.

If INC is positive, the selected starting value (FIRSTV) approximates a minimum, and the scale factor (DELTAV) is positive.

If INC is negative, the selected starting value (FIRSTV) approximates a maximum, and the scaling factor (DELTAV) is negative.

If INC = $\pm$1, the array must be dimensioned at least two elements larger than the actual number of data values it contains. If the magnitude of INC is greater than 1, the computed values are stored at (INC) elements and (2*INC) elements beyond the last data point. The subscripted element for FIRSTV is ARRAY (NPTS*INC+1), and for DELTAV is ARRAY (NPTS*INC+INC+).

FIRSTV: is the starting value, which will appear at the first tick mark origin, on the axis. This value will be specified by the model input data file in order to standardize plotting scales while still permitting some flexibility to look at some plots in more detail.

This number and scale value among the axis is drawn with two decimal places. Since the digit size is about 10 characters per inch, and since value appears every inch, no more than six digits and a sign would appear to the left of the decimal point.

DELTAV: represents the number of data units per inch of axis. This value (increment or decrement), which is added to FIRSTV for each succeeding 1-inch division along the axis, will also be specified in the model input data file.

APPENDIX B

IBM/MVS/TSO INSTALLATIONS

IBM MVS/TSO INSTALLATION.

The following setup provides a basic outline to move the first file from the input tape to the target MVS system. It would be likely that this setup would have to have minor changes made to it in order to satisfy system dependent differences. In this example, the first file is being copied to a disk file named USERID.BMLST.FORT.

To retrieve a file on magnetic tape:

```
//NAME    JOB   ...
//STEP1   EXEC  PGM=IEBGENER
//SYSUT1    DD  DSN=DUMMY,UNIT=TAPE,LABEL=(1,BLP),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200,DEN=3,OPTCD=Q)
//SYSUT2    DD  DSN=USERID.BMLST.FORT, DISP=(NEW,CATLG),
//      DCB=*.SYSUT1,SPACE=(CYL,(1,1)),UNIT=DISK
//SYSIN     DD  DUMMY
//SYSPRINT  DD  SYSOUT=A
```

The following similar setup could be used to access the second file and copy it to direct access storage with the name USERID.BMLSR.FORT. The only difference being in the LABEL field and the DSN change to give the data set a different name and specify the second file.

```
//NAME    JOB   ...
//STEP1   EXEC  PGM=IEBGENER
//SYSUT1    DD  DSN=DUMMY,UNIT=TAPE,
//      LABEL=(2,BLP),DCB=(RECFM=FB,LRECL=80,OPCTD=Q,
//      BLKSIZE=3200,DEN=3)
//SYSUT2    DD  DSN=USERID.BMLSR.FORT, DISP=(NEW,CATLG),
//      DCB=*.SYSUT1,SPACE=(CYL,(1,1)),UNIT=DISK
//SYSIN     DD  DUMMY
//SYSPRINT  DD  SYSOUT=A
```

Using this general approach all the data sets on the unlabeled tape can be copied to direct access storage. Appropriate names should be chosen for the sample input and output data sets, as these will be required when the model is executed.

COMPILATION AND LINKING.

After the tape has been copied to disk, a compile and link process will be required. All large IBM installations have a group of "cataloged procedures" that assist in compiling source code into object code and then "linking" the object into an executable image. The system manager will be able to supply appropriate JCL statements that can be used to accomplish this. These procedures can also be modified so as to create a load module that can be executed without repeating the compilation and linking steps. Other procedures are available or can be coded to execute the load module against the sample inputs that have been copied to disk.

The graphics software must be input in the link step of the catalogued procedure provided. The user may choose any name for this library when it is created.

APPENDIX C

DEC/VAX 11/750/780 INSTALLATIONS

DIGITAL EQUIPMENT CORP. (DEC) VAX 11/780/50; MICROVAX II.

After logging into the VAX, create a directory in your USER ID to hold the source code from the tape. Assuming your USER ID is "PAULA" then, at the $ prompt, type:

```
$ CREATE/DIRECTORY [PAULA.MODEL]
```

Then make this your "default directory" with:

```
$ SET DEFAULT [PAULA.MODEL]
```

Next reserve a tape drive for the transmittal tape with:

```
$ ALLOCATE MT:
```

The name MT: is the device name of the tape drive you want to use. It may be different on your installation - check with the system administrator. Use whatever name is suggested instead of MT: here and anywhere else MT: appears in this discussion.

At this point you must physically mount the tape on the drive. Do NOT put a "write ring" in the tape. That will safeguard the data on the tape from accidental erasure. Some tape drives allow you to take the same precaution with push buttons on the front panel.

Next mount the volume with:

```
$ MOUNT/FOREIGN/NOWRITE/BLOCK=3200/DENSITY=1600/RECORD=80 MT:
```

Next copy the first file (which contains source code) to disk by typing:

```
$ COPY MT: BMLST.FOR
```

Copy the second file in the same way just use a different name:

```
$ COPY MT: BMLSR.FOR
```

Next copy the two plotting source files with:

```
$ COPY MT: BPLOTT.FOR
$ COPY MT: BPLOTR.FOR
```

Next copy the input files to disk by a series of copies like:

```
$ COPY MT: INPUT1
$ COPY MT: INPUT2
$ COPY MT: INPUT3
$ COPY MT: INPUT4
$ COPY MT: INPUT5
```

Enter as many copy commands as necessary to retrieve all the input data sets. If the distribution notes specify, copy all succeeding files to disk choosing appropriate names for the files.

This completes the transfer of data to disk. You can release the tape drive by these commands and then unload the tape:

```
$ DISMOUNT MT:
$ DEALLOCATE MT:
```

The programs must now be compiled and linked to create an ".exe" file that can be executed. To compile the programs type the following commands:

```
$ FORT BMLST
$ FORT BMLSR
$ FORT BPLOTT
$ FORT BPLOTR
$ FORT USERPLOT
$ FORT SINIT
$ FORT SOPEN
$ LINK BMLST, SINIT, SOPEN
$ LINK BMLSR, SINIT, SOPEN
$ LINK BPLOTT, USERPLOT, SINIT, SOPEN
$ LINK BPLOTR, USERPLOT, SINIT, SOPEN
```

Your system may have a different command than "FORT", e.g., "F77" or "FORTRAN" to invoke the FORTRAN 77 compiler. Check with the system administrator if "FORT" doesn't work.

You must provide a plot routine library called "USERPLOT.OBJ" for input to the linkage editor. See appendix A for specifications of the source code.

This completes the basic installation procedure. To exercise the model refer to the user's manual to select appropriate input data sets and run the simulation.

APPENDIX D

UNIX SYSTEM V SYSTEMS INSTALLATIONS

UNIX INSTALLATIONS.

The installation of the MLS model on a UNIX System V computer requires that the transmittal tape be post-processed to generate disk data sets that are suitable for compilation and editing. A short "C" program is included here to assist in reformatting the source code and input data sets after they have been copied to disk.

The following program listing may prove useful to the installer:

```
/* sample program */
#include K stdio.h k
main()
?
FILE *fpr;
int bal[80], *buffer;
buffer = bal;
fpr=fopen("tran","r");

while ( ! feof(fpr)  )
?
  fgets( buffer, 81, fpr);
      printf("%s : n",bal);
/
/
```

This program inserts a new line character (hex 0a) after each 80 characters in the file "tran." The output is written to standard output so it can be redirected.

This program may be compiled in the usual UNIX fashion with:

```
$ cc name.c
```

Where "name.c" was the name given to the above program when created.

If all goes well this will produce the default executable file "a.out."

Copying files:

The tape files can be copied by using the standard "dd" utility.

The following would copy the first file from the tape:

```
$ dd if=/dev/rmt1 of=BMLST ibs=3200
```

Where "/dev/rmt1" would most likely have to be changed to specify your installation's 9-track drive at 1600 bpi. This will create a file called "BMLST" in your default directory. Subsequent files on the tape can be copied in a similar manner, keeping in mind that you must pick a unique name for the output file each time. You will also have to understand how to keep the tape from rewinding on close, or how to space to the desired file each time you copy a file. No rewind may be the default on your system, or you may have to specify a different device name to get no rewind. Best advice is to check with someone about this, if in doubt.

Assuming that each file has been successfully copied, they may be reformatted using the following procedure:

```
$ mv BMLST tran

$ a.out k temp

$ mv temp BMLST
```

This will put the new line characters into the files so that they can be compiled and, if necessary, edited. These three statements need to be repeated for each filename that was copied to disk.

The next step is to compile the source code files to get executable files. The standard FORTRAN 77 compiler command is "F77." The following command could be used. However, if the compiler command has a different name, substitute that for "F77" in this example.

```
$ F77 BMLST SINIT SOPEN

$ mv a.out BMLST.o

$ F77 BMLSR SINIT SOPEN

$ mv a.out BMLSR.o

$ F77 BPLOTT SINIT SOPEN USERPLOT

$ mv a.out BPLOTT.o

$ F77 BPLOTR SINIT SOPEN USERPLOT

$ mv a.out BPLOTR.o
```

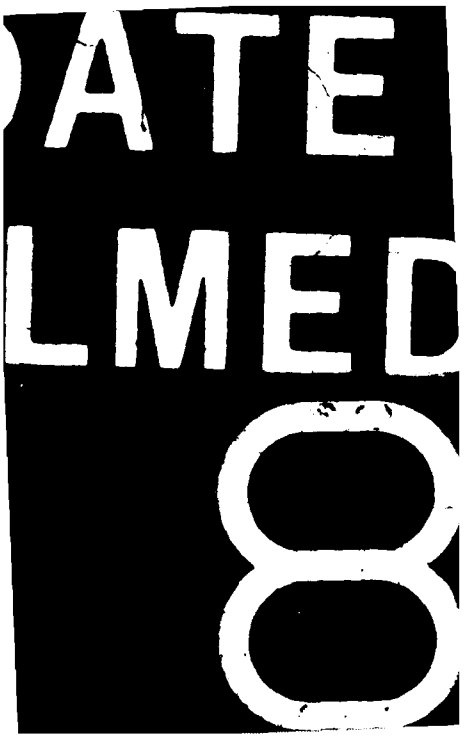
This will create the necessary load files to execute the simulation and execute the plotting programs. Remember that USERPLOT is a FORTRAN 77 program that the user site must provide. SINIT and SOPEN are initialization module provided for user modification when the model was first delivered.

For the curious, "mv" is the UNIX rename command. The residual files with the ".o' suffix are executable files. To execute any of the four programs just type the file name. For example:

```
$ BPLOTT.O
```

will involve the BPLOTT software.

You must set up the appropriate input files before invoking this program or it will terminate with errors.


ATE
LMED
8